Expert Systems with Applications 37 (2010) 4761–4767

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# An artificial bee colony approach for clustering

Changsheng Zhang [a,*], Dantong Ouyang [b], Jiaxu Ning [c]

[a] College of Information Science & Engineering, Northeastern University, Shenyang 110819, PR China
[b] Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, PR China
[c] Institute of Grassland Science Northeast Normal University, PR China

ARTICLE INFO

*Keywords:*
Clustering
Meta-heuristic algorithm
Artificial bee colony
*K*-means

ABSTRACT

Clustering is a popular data analysis and data mining technique. In this paper, an artificial bee colony clustering algorithm is presented to optimally partition *N* objects into *K* clusters. The *Deb's* rules are used to direct the search direction of each candidate. This algorithm has been tested on several well-known real datasets and compared with other popular heuristics algorithm in clustering, such as GA, SA, TS, ACO and the recently proposed K–NM–PSO algorithm. The computational simulations reveal very encouraging results in terms of the quality of solution and the processing time required.



## 1. Introduction

Clustering is an important problem that must often be solved as a part of more complicated tasks in pattern recognition, image analysis and other fields of science and engineering. Clustering procedures partition a set of objects into clusters such that objects in the same cluster are more similar to each other than objects in different clusters according to some predefined criteria. The existing clustering algorithms can be simply classified into the following two categories: hierarchical clustering and partitional clustering (Sander, 2003.). Hierarchical clustering operates by partitioning the patterns into successively fewer structures. Since it is not the subject of this study we will not mention it in detail. Partitional clustering procedures typically start with the patterns partitioning into a number of clusters and divide the patterns by increasing the number of partitions. The most popular class of partitional clustering methods is the center-based clustering algorithms.

*K*-means has been used as a popular center-based clustering method due to its simplicity and efficiency, with linear time complexity. However, *K*-means has the shortcomings of depending on the initial state and converging to local minima (Selim & Ismail, 1984). In order to overcome these problems, many heuristic clustering algorithms have been introduced. A genetic algorithm based method to solve the clustering problem was proposed by Mualik and Bandyopadhyay (2002) and experimented on synthetic and real-life datasets to evaluate its performance. Krishna and Murty (1999) proposed a novel approach called genetic *K*-means algorithm for clustering analysis which defines a basic mutation operator specific to clustering called distance-based mutation. It has been proved that GKA converge to the best-known optimum through using the theory of finite Markov chain. A simulated annealing approach for solving the clustering problem is proposed by Selim and Al-Sultan (1991). The parameters of the algorithm were discussed in detail and it has been proved theoretically that a clustering problem's global solution can be reached. Sung and Jin (2000) proposed a tabu search based heuristic for clustering. Two complementary functional procedures, called packing and releasing procedures were combined with the tabu search.

Over the last decade, modeling the behavior of social insects, such as birds, ants, and bees for the purpose of search and optimization has become an emerging area of swarm intelligence and successfully applied to clustering. An ant colony clustering algorithm for clustering is presented by Shelokar, Jayaraman, and Kulkarni (2004). The algorithm employs distributed agents who mimic the way real ants find a shortest path from their nest to food source and back. Its performance was compared with GA, tabu search, and SA. The particle swarm optimization which simulates bird flocking was used for clustering by Kao, Zahara, and Kao (2008). In order to improve its performance further, the PSO algorithm is hybridized with *K*-means and Nelder–Mead simplex search method. Its performance is compared with GA (Murthy & Chowdhury, 1996) and KGA (Bandyopadhyay & Maulik, 2002) algorithm.

Honey-bees are among the most closely studied social insets. Their foraging behavior, learning, memorizing and information sharing characteristics have recently been one of the most interesting research areas in swarm intelligence (Teodorovic et al., 2006). Recently, Karaboga and Basturk (2008) have described an artificial bee colony (ABC) algorithm based on the foraging behavior of honey-bees for numerical optimization problems. They have compared the performance of the ABC algorithm with those of other well-known modern heuristic algorithms such as genetic algorithm, differential evolutional algorithm and particle swarm

* Corresponding author. Tel.: +86 0431 85166487.
*E-mail address:* zcs820@yahoo.com.cn (D. Ouyang).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.11.003

optimization algorithm for unconstrained optimization problems. In this work, ABC algorithm is extended for solving clustering problems. The performance of the algorithm has been tested on a variety of data sets provided from several real-life situations and compared with several other proposed clustering algorithms. This paper is organized as follows. In Section 2, we discussed the clustering analysis problems. The ABC algorithm and the ABC algorithm adapted for solving clustering problems are introduced in Section 3. Section 4 will present experimental studies that show that our method outperforms some other methods. Finally, Section 5 summarizes the contribution of this paper along with some future research directions.

## 2. The clustering problem

Let $\boldsymbol{O} = \{o_1, o_2, \ldots, o_n\}$ be a set of $n$ objects and let $\boldsymbol{X}_{n\times p}$ be the profile data matrix, with $n$ rows and $p$ columns. Each $i$th objects is characterized by a real-value $p$-dimensional profile vector $x_i (i = 1, \ldots, n)$, where each element $x_{ij}$ corresponds to the $j$th real-value feature ($j = 1, \ldots, p$) of the $i$th object ($i = 1, \ldots, n$).

Given $X_{n\times p}$, the goal of a partitional clustering algorithm is to determine a partition $G = \{C_1, C_2, \ldots, C_k\}$ (*i.e.*, $C_g \neq \Phi,\ \forall g;\ C_g \cap C_h = \Phi,\ \forall g \neq h;\ \cup_{g=1}^{k} C_g = O$) such that objects which belong to the same cluster are as similar to each other as possible, while objects which belong to different clusters are as dissimilar as possible. For this, a measure of adequacy for the partition must be defined. A popular function used to quantify the goodness of a partition is the total within-cluster variance or the total mean-square quantization error (MSE) (Güngör & Ünler, 2007 ) which is defined as follows:

$$\text{Perf}\ (O, G) = \sum_{i=1}^{n} \text{Min}\left\{ \|o_i - C_l\|^2 | l = 1, \ldots k \right\} \tag{1}$$

Where $\|o_i - C_l\|$ denotes the similarity between object $o_i$ and center $C_l$. The most used similarity metric in clustering procedure is Euclidean distance which is derived from the Minkowski metric.

$$d(x, y) = \left( \sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r} \Rightarrow d(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{2}$$

In this study we will also use Euclidean metric as a distance metric. The clustering problem is to find the partition $G^*$ that has optimal adequacy with respect to all other feasible solutions $G = \{G^1, G^2, \ldots, G^{N(n,k)}\}$ (*i.e.*, $G^i \neq G^j,\ i \neq j$) where

$$N(n, k) = \frac{1}{k!} \sum_{g=0}^{k} (-1)^g \binom{k}{g} (k - g)^n$$

It is the number of all feasible partitions. It has been shown that the clustering problem is NP-hard when the number of clusters exceeds three (Brucker, 1978).

## 3. Artificial bee colony based clustering

### 3.1. Honey bee modeling (Karaboga & Basturk, 2008)

The minimal model of forage selection that leads to the emergence of social intelligence of honey bee swarms consists of three essential components: food sources, employed foragers and unemployed foragers, and two leading modes of the behavior, recruitment to a nectar source and abandonment of a source, are defined (Karaboga, 2005). A food source value depends on many factors, such as its proximity to the nest, richness or concentration of energy and the ease of extracting this energy. The employed foragers are associated with particular food sources, which they are currently exploiting or are "employed". They carry with them information about these food sources and share this information with a certain probability. There are two types of unemployed foragers, scouts and onlookers. Scouts search the environment surrounding the nest for new food sources, and onlookers wait in the nest and find a food source through the information shared by employed foragers.

In ABC algorithm (Basturk & Karaboga, 2006; Karaboga & Basturk, 2008), the colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts. A food source represents a possible solution to the problem to be optimized. The nectar amount of a food source corresponds to the quality of the solution represented by that food source. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source has been abandoned by the bees becomes a scout.

As other social foragers, bees search for food sources in a way that maximizes the ration $E/T$ where $E$ is the energy obtained and $T$ is the time spent for foraging. In the case of artificial bee swarms, $E$ is proportional to the nectar amount of food sources discovered by bees. In a maximization problem, the goal is to find the maximum of the objective function $F(\theta), \theta \in R^P$. Assume that $\theta_i$ is the position of the $i$th food source; $F(\theta_i)$ represents the nectar amount of the food source located at $\theta_i$ and is proportional to the energy $E(\theta_i)$. Let $P(c) = \{\theta_i(c) | i = 1, 2, \ldots, S\}$ ($c$: cycle, $S$: number of food sources being visited by bees) represent the population of food sources being visited by bees.

As mentioned above, the preference of a food source by an onlooker depends on the nectar amount $F(\theta)$ of that food source. As the nectar amount of the food source increases, the probability with the preferred source by an onlooker bee increases proportionally. Therefore, the probability with the food source located at $\theta_i$ will be chosen by a bee can be calculated as

$$P_i = \frac{F(\theta_i)}{\sum_{k=1}^{S} F(\theta_k)} \tag{3}$$

After watching the dances of employed bees, an onlooker bee goes to the region of food source located at $\theta_i$ by this probability and determines a neighbor food source to take its nectar depending on some visual information, such as signs existing on the patches. In other words, the onlooker bee selects one of the food sources after making a comparison among the food sources around $\theta_i$. The position of the selected neighbor food source can be calculated as $\theta_i(c + 1) = \theta_i(c) \pm \phi_i(c)$. $\phi_i(c)$ is a randomly produced step to find a food source with more nectar around $\theta_i$. $\phi(c)$ is calculated by taking the difference of the same parts of $\theta_i(c)$ and $\theta_k(c)$ ($k$ is a randomly produced index) food positions. If the nectar amount $F(\theta_i(c + 1))$ at $\theta_i(c + 1)$ is higher than that at $\theta_i(c)$, then the bee goes to the hive and share her information with others and the position $\theta_i(c)$ of the food source is changed to be $\theta_i(c + 1)$, otherwise $\theta_i(c)$ is kept as it is.

Every food source has only one employed bee. Therefore, the number of employed bees is equal to the number of food sources. If the position $\theta_i$ of the food source $i$ cannot be improved through the predetermined number of trials "*limit*", then that food source $\theta_i$ is abandoned by its employed bee and then the employed bee becomes a scout. The scout starts to search a new food source, and after finding a new source, the new position is accepted to be $\theta_i$. Every bee colony has scouts that are the colony's explorers. The explorers do not have any guidance while looking for food. They are primarily concerned with finding any kind of food source. As a result of such behavior, the scouts are characterized by low search costs and a low average in food source quality. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. In the case of artificial bees, the artificial scouts could have the fast discovery of the group of feasible solutions as a task.

It is clear from the above explanation that there are four control parameters used in the ABC algorithm: the number of food sources which is equal to the number of employed bees ($S$), the value of "*limit*" and the maximum cycle number ($MCG$). The main steps of the algorithm can be described as follows:

*Step 1:* Initialize the population of solutions. $\theta_i,\ i = 1, \ldots, S$ and evaluate them.
*Step 2:* Produce new solutions for the employed bees, evaluate them and apply the greedy selection process.
*Step 3:* Calculate the probabilities of the current sources with which they are preferred by the onlookers.
*Step 4:* Assign onlooker bees to employed bees according to probabilities, produce new solutions and apply the greedy selection process.
*Step 5:* Stop the exploitation process of the sources abandoned by bees and send the scouts in the search area for discovering new food sources, randomly.
*Step 6:* Memorize the best food source found so far.
*Step 7:* If the termination condition is not satisfied, go to *step 2*, otherwise stop the algorithm.

After each candidate source position being produced and evaluated by the artificial bee, its performance is compared with that of its old one. If the new food has an equal or better nectar amount than the old one, it is replaced with the old one in the memory. Otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one. Furthermore, the mean number of scouts averaged over conditions is about 5–10% (Karaboga & Basturk, 2008).

### 3.2. The ABC algorithm used for clustering problems

From the Section 3.1, we know that there exists a population of individuals (bees) in the ABC algorithm. Each individual consists of an encoding of a candidate solution (food source) and a fitness that indicates its quality. In order to apply it to solve clustering problem, we have used floating point arrays to encode cluster centers. Hence, if $X_{n\times p}$ is the profile matrix and $k$ the number of clusters $G = \{C_1, C_2, \ldots, C_k\}$ of the set of $n$ objects $O = \{o_1, o_2, \ldots, o_n\}$, each candidate solution in the population consists of $p$ times $k$ cells $m_{ij} (i \in \{1, \ldots, k\},\ j \in \{1, \ldots, p\})$. Each group of $p$ cells that corresponds to the vector $m_i$, represents the coordinates of the $i$th cluster center. The $k$ groups of $p$ cells that constitute the vector $m$ represent the $k$ cluster centers. Fig. 1 shows an example for problem with four clusters and four features.

A set of $k$ cluster centers specifies an objects partition by mapping the cluster search space to the partition search space$G = \{G^1, G^2, \ldots, G^{N(n,k)}\}$. The mapping is inspired by *Forgy's* approach of clustering (Forgy, 1965) in which a partition is determined by allocating each object to the cluster that is associated with its nearest cluster center. "Nearest" refers to distance metric, which is the Euclidean distance in out study. According to the description of clustering problem in the Section 2, a feasible solution must satisfied the following three conditions: $C_g \neq \Phi,\ g \in \{1, \ldots, k\}; \cup_{g=1}^{k} C_g = O;\ C_g \cap C_h = \Phi,\ g \neq h,\ g \in \{1, \ldots, k\},\ h \in \{1, \ldots, k\}$. We can see that if the first condition is satisfied, the latter two conditions would also be satisfied since each object in $\boldsymbol{O}$ is assigned to its nearest cluster center.

To tackle the infeasible solutions, we adopted *Deb's* constrained handling method (Goldberg & Deb, 1991) instead of the greedy selection process of the ABC algorithm described in the previous section since *Deb's* method consists of very simple three heuristic rules. *Deb's* method uses a tournament selection operator, where two solutions are compared at a time, and the following criteria

| | | | | | |
|---|---|---|---|---|---|
| $m_1$ | $m_{11}$ | $m_{12}$ | $m_{13}$ | $m_{14}$ | *centroid coordinates of cluster 1* |
| $m_2$ | $m_{21}$ | $m_{22}$ | $m_{23}$ | $m_{24}$ | *centroid coordinates of cluster 2* |
| $m_3$ | $m_{31}$ | $m_{32}$ | $m_{33}$ | $m_{34}$ | *centroid coordinates of cluster 3* |
| $m_4$ | $m_{41}$ | $m_{42}$ | $m_{43}$ | $m_{44}$ | *centroid coordinates of cluster 4* |

**Fig. 1.** Example of a candidate solution encoding with four clusters and four features.

are always enforced. Any feasible solution is preferred to any infeasible solution; among two feasible solutions, the one having better objective function value is preferred; among two infeasible solutions, the one having smaller constraint violation is preferred.

In order to produce a candidate food position $\theta^q$ from the current memorized $q$th source position $C^q$, the adapted ABC algorithm uses the following expression:

$$\theta_{ij}^q = \begin{cases} C_{ij}^q + \phi_{ij}(C_{i,j}^q - C_{i,j}^r), & \text{if } R_j < MR \\ C_{ij}^q, & \text{otherwise} \end{cases} \tag{4}$$

where $r \in \{1, \ldots, SN\}$ is an randomly chosen index, and $j \in \{1, \ldots, p\}, i \in \{1, \ldots, k\}$. Although $r$ is determined randomly, it has to be different from $q$. $\phi_{ij}$ is a random number between $[-1, 1]$. It controls the production of neighbor food sources around $C^q$ and represents the comparison of two food positions visually by a bee. $R_j$ is randomly chosen real number in the range $[0, 1]$. MR, modification rate, is a control parameter that controls whether the element $C_{ij}^q$ will be modified or not. As can be seen from (4), as the difference between the elements of the $C^q$ and $C^r$ decreases, the perturbation on the position $C^q$ gets decrease, too. Thus, as the search approaches to the optimum solution in the search space, the step length is adaptively reduced. If an element value produced by this operation exceeds its predetermined limit, the element can be set to an acceptable value. In this work, the value of the element exceeding its *limit* is set to its *limit* value.

In the real bee colony, the employed bee whose food source has been exhausted by the bees becomes a scout. If a scout discovered a rich food source, it would be employed. In order to simulate this behavior of real bees, the following strategy is used in this paper: providing that a position can not be improved further through a predetermined number of cycles, the food source is assumed to be abandoned and the corresponding employed bee becomes a scout for exploration. When the number of scouts reaches a predetermined upper bound, the employed bees and scouts will be ordered together according to their found food source qualities and make the bees with worst food sources as scouts and others as employed bees. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called "*limit*" for abandonment. As the colony's explorers (Seeley & Visscher, 1988), the scouts do not have any guidance while looking for food. In the ABC algorithm used in this paper, this is simulated by producing a position $\theta$ randomly as follows:

$$\theta_{ij} = \theta_j^{\min} + r \text{ and } (0,1)(\theta_j^{\max} - \theta_j^{\min}); \quad i \in \{1, \ldots, p\}, \quad j \in \{1, \ldots, k\} \tag{5}$$

where $\theta_j^{\min}$ and $\theta_j^{\max}$ are the minimum and maximum values of the $j$th object feature. In principle any point in $R^p$ could be considered as a possible choice for a cluster center. It is usually chosen to be the profile matrix domain $[\theta^{\min}, \theta^{\max}]$, where $\theta^{\min}$ and $\theta^{\max}$ are two vectors characterizing the minimum and maximum object values found in the data set for each feature. However, in our study, the cluster domain is 40% larger because good cluster centers could lie beyond the profile matrix domain border.

Based on the above descriptions, the flowchart of the ABC algorithm used in this paper is shown as Fig. 2. It is clear that there exist four control parameters used in this algorithm: the swarm size *N*, the upper bound *UB* which regulates the maximum permitted number of scouts, the value of *limit* and the maximum cycle number *MCN*. Detailed description of each step is given below:

*Step 1: Initialization*
Set the control parameter values. Make the first half of the colony consists of the employed bees and the second half includes the onlookers. Then randomly generate a position for each candidate and evaluate it. Set the current scout number $s = 0$.

*Step 2: Introduce new food sources discovered by scouts*
If $s > UB$, order the first half of colony, make the bees with worst solution quality as scouts and others as employed bees. Update the scout number *s*.

*Step 3: Employed bees exploitation*
Produce new solution for each employed bee by using (4) and evaluate it. Then the selection process by using *Deb's* method is applied. If the "*limit*" for abandonment is reached, the employed bee forgets its memory and becomes a scout for exploration. The scouts number $s = s + 1$.

*Step 4: Scouts exploration*
Send each scout into the search area for discovering new food sources randomly by using (5). When a new food source is found, evaluate it and the selection process of *Deb's* method is applied.

*Step 5: Preferences computation for the current food sources*
Calculate the probability values of the current food sources with which they are preferred by the onlookers according to Eq. (3).

*Step 6: Onlookers exploitation*
For the onlookers, produce new solutions from the current food sources selected depending on the computed probabilities and evaluate them. Then the selection process by using *Deb's* method is applied to update the corresponding employed bee's memory or the current food sources.

*Step 7: Memorize the best position*
For each employed bee and scout, if its memorized position is better than the previous achieved best position, then the best position is replaced by it.

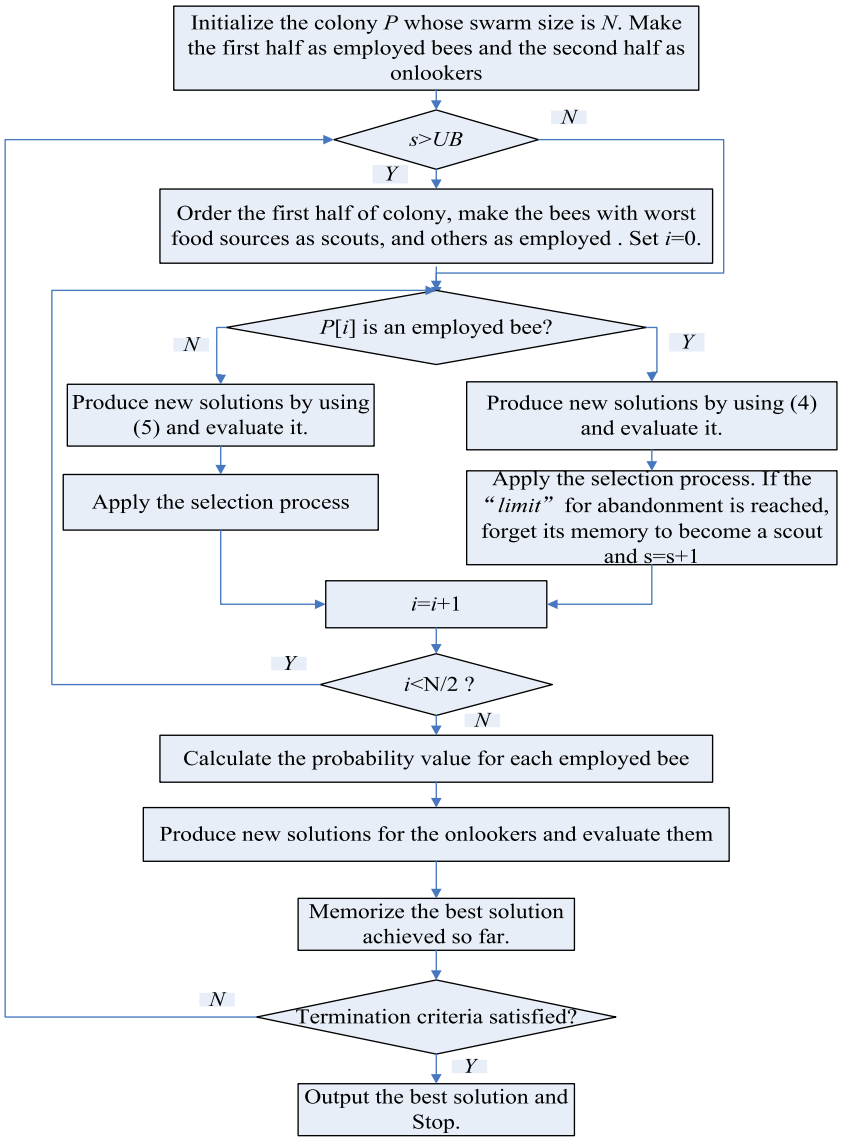


**Fig. 2.** The flow chart of ABC algorithm used for clustering.

*Step 8: Check the termination criteria*
If the termination condition is not satisfied, go to *step 2*, otherwise stop the algorithm.

Because initialization with feasible solutions is very time consuming and in some cases it is impossible to produce a feasible solution randomly, the ABC algorithm does not consider the initial population to be feasible. Structure of the algorithm already directs the solutions to feasible region in running process due to the *Deb's* rules employed instead of greedy selection. Scout production process of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population.

## 4. Results and discussion

We test the ABC clustering algorithm on three different scale datasets and compared with other well-known algorithms. All algorithms are implemented in C++ language and executed on a Pentium IV, 2.8HZ, 512GB RAM computer. The three datasets are well-known iris, thyroid, and wine datasets taken from Machine Learning Laboratory (Blake & Merz, 1998). They have been considered by many authors to study and evaluate the performance of their algorithms, and can be described as follows:

*Data set 1: The Iris dataset.* It is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. The data set contains three categories of 50 objects each, where each category refers to a type of iris plant. One category is linearly separable from the other two; the latter are not linearly separable from each other. There are 150 instances with four numeric features in iris data set. There is no missing attribute value. The attributes of the iris data set are sepal length in cm, sepal width in cm, petal length in cm and petal width in cm.

*Data set 2: The thyroid gland dataset.* This data set contains 215 samples of patients suffering from three human thyroid diseases: euthyroidism, hypothyroidism and hyperthyroidism where 150 individuals are tested euthyroidism thyroid, 30 patients are experienced hyperthyroidism thyroid while 35 patients are suffered from hypothyroidism thyroid. Each individual was characterized by five features of laboratory tests: T3-resin uptake test, total Serum thyroxin as measured by the isotopic displacement method, total serum triiodothyronine as measured by radioimmuno assay, basal thyroid-stimulating hormone (TSH) as measured by radioimmuno assay, maximal absolute difference of TSH value after injection of 200 micro grams of thyrotropin releasing-hormone as compared to the basal value.

*Data set 3: The wine dataset.* This dataset contains chemical analysis of 178 wines, derived from three different cultivars. Wine type is based on 13 continuous features derived from chemical analysis: alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavanoids, nonflavanoid phenols, proanthocyaninsm, color intensity, hue, OD280/OD315 of diluted wines and praline. The quantities of objects in the three categories of the data set are 59, 71 and 48, respectively.

To evaluate the performance of the ABC algorithm, we have compared it with the following clustering algorithms: GA (Murthy & Chowdhury, 1996), TS (Al-Sultan, 1995), SA (Selim & Al-Sultan, 1991), ACO (Shelokar et al., 2004), K–NM–PSO (Kao et al., 2008.). There are four control parameters in ABC algorithm, the swarm size *N*, the upper bounce *UB*, the "*limit*" and the maximum cycle number *MCN*. They are set as follows: $N = 20$, $UB = 5$, $limit = 10$, $MCN = 2000$. The parameter settings of ACO, GA, TS, SA and K–NM–PSO are set the same as their original paper. The sum of the intra-cluster distances, i.e. the distances between data vectors within a cluster and the centroid of this cluster, as defined in Eq. (1) is used to measure the quality of a clustering. Clearly, the smaller the sum of the distances is, the higher the quality of clustering.

The effectiveness of stochastic algorithms is greatly dependent on the generation of initial solutions. Therefore, for every dataset, algorithms performed 10 times individually for their own effectiveness tests, each time with randomly generated initial solutions. Table 1 summarizes the intra-cluster distances obtained from the six clustering algorithms for the data sets above. The values reported are averages of the sums of intra-cluster distances and the fitness values of the worst and best solutions which can indicate the range of values that the algorithms span.

From the Table 1, we can see that the ABC algorithm has achieved the best performance in terms of the average, best, and worst inter-cluster distances on these three data sets. For Iris dataset, the best intra-cluster distance obtained by GA is worst which is 113.98 and the ABC algorithm provides the optimum value of 78.94, greatly better than other compared algorithms. Moreover, the worst intra-cluster distance obtained by ABC algorithm is also 78.94, which indicates that it is able to find the optimum every time. The centroids coordinates for the best are show in Table 2 and the corresponding three dimension clustering result is given in Fig. 3 which can make it visualized clearly from different views. For the other two data sets, the TS algorithm performs worse than the GA algorithm and the hybrid K–NM–PSO algorithm is only inferior to the ABC algorithm, but better than other compared algorithms in term of solution qualities. Furthermore, the average number of function evaluations required and the average processing time taken to attain the best solution for each algorithm is also compared and given in Table 3. For both the Iris dataset and

**Table 2**
The achieved best centroids coordinates for iris data.

| | Feature A | Feature B | Feature C | Feature D |
|---|---|---|---|---|
| Cluster 1 | 5.9016137 | 2.748387 | 4.393549 | 1.4338713 |
| Cluster 2 | 5.005996 | 3.4180002 | 1.464 | 0.24399997 |
| Cluster 3 | 6.85 | 3.073684 | 5.7421055 | 2.0710523 |

**Table 1**
Comparison of intra-cluster distances for the six algorithms.

| Data set | Criteria | GA | TS | SA | ACO | K–NM–PSO | ABC |
|---|---|---|---|---|---|---|---|
| Iris | Average | 125.19 | 97.86 | 97.13 | 97.17 | 96.67 | 78.94 |
| | Worst | 139.78 | 98.57 | 97.26 | 97.81 | 97.01 | 78.94 |
| | Best | 113.98 | 97.36 | 97.10 | 97.10 | 96.66 | 78.94 |
| Thyroid | Average | 10128.82 | 10354.31 | 10114.04 | 10112.13 | 10109.70 | 10104.03 |
| | Worst | 10148.39 | 10438.78 | 10115.93 | 10114.82 | 10112.86 | 10108.24 |
| | Best | 10116.29 | 10249.73 | 10111.82 | 10111.82 | 10108.56 | 10100.31 |
| Wine | Average | 16530.53 | 16785.46 | 16530.53 | 16530.53 | 16293.00 | 16260.52 |
| | Worst | 16530.53 | 16837.54 | 16530.53 | 16530.53 | 16295.46 | 16279.46 |
| | Best | 16530.53 | 16666.22 | 16530.53 | 16530.53 | 16292.00 | 16257.28 |

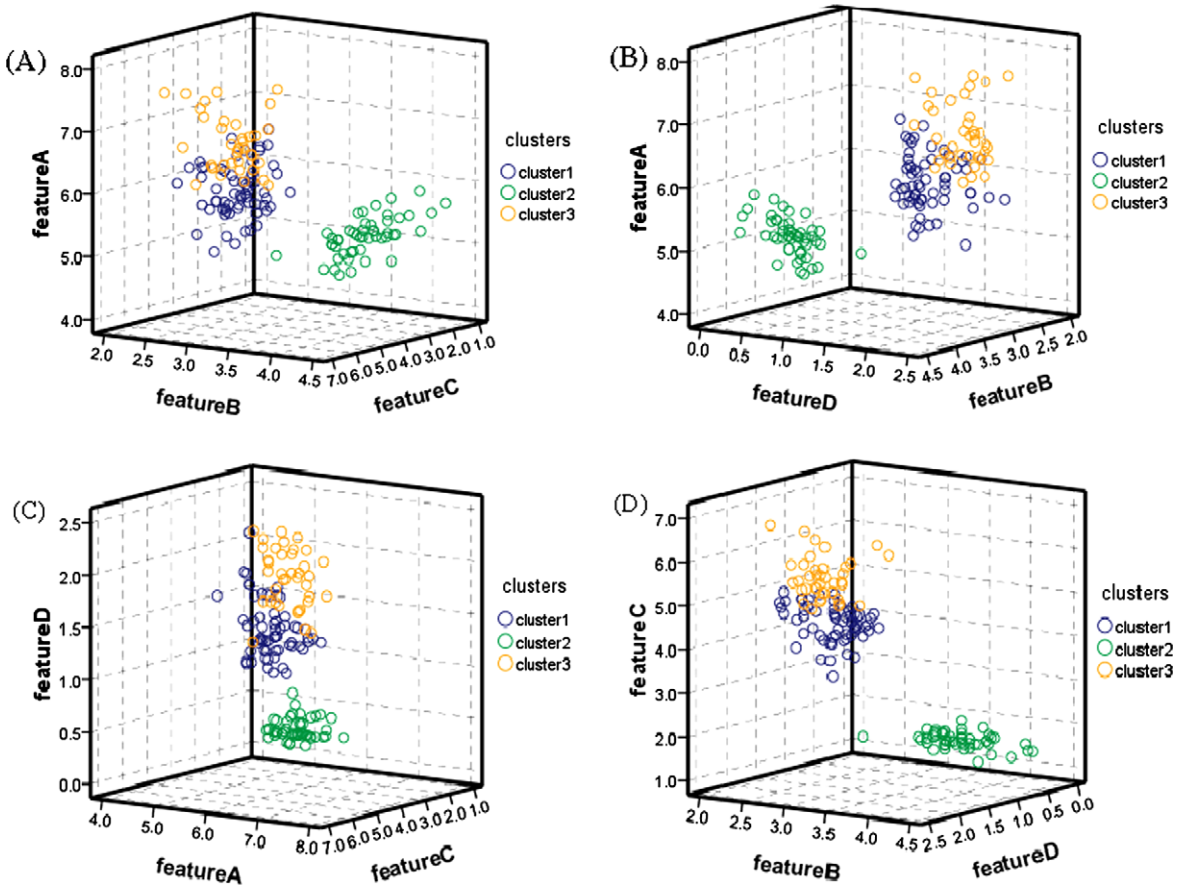


**Fig. 3.** Clustering result of Iris data by ABC algorithm.

**Table 3**
The average fitness computation numbers and computation time.

| Data set | | GA | TS | SA | ACO | K–NM–PSO | ABC |
|---|---|---|---|---|---|---|---|
| Iris | Time (s) | 105.53 | 72.86 | 95.92 | 33.72 | 48.13 | **29.68** |
| | Numbers | 38128 | 20201 | 29103 | 10998 | **4556** | 8658 |
| Thyroid | Time (s) | 153.24 | 114.01 | 108.22 | 102.15 | 118.46 | **85.26** |
| | Numbers | 45003 | 29191 | 28675 | 25626 | **7245** | 24136 |
| Wine | Time (s) | 226.68 | 161.45 | 57.28 | 68.29 | 589.40 | **48.85** |
| | Numbers | 33551 | 22716 | **7917** | 9306 | 46459 | 17554 |

thyroid dataset, the GA algorithm consumes the most processing time and fitness evaluation numbers, and the ABC algorithm needs the least processing times which are 29.68 and 85.26 s, respectively, but it takes more fitness evaluation numbers than the K–NM–PSO algorithm. However, the K–NM–PSO algorithm obviously consumes the most processing time and fitness evaluations, and the ABC algorithm costs the least processing time on the wine dataset. This is mainly for that during each generation, the candidates of the swarm are sorted by fitness and a local search process is executed for some particles in K–NM–PSO algorithm. According to its parameter setting rules, the swarm size is 118 for wine dataset, and much time is consumed by the rank process.

From the above results, we can obtain that the ABC algorithm performed better than other compared algorithms in terms of processing time and intra-cluster distance. Its superiority is evident and can be considered as a viable and an efficient heuristic to find optimal or near optimal solutions to clustering problems of allocating $N$ objects to $K$ clusters.

## 5. Conclusions

Modeling the behavior of social insects, such as ants, birds or bees, for the purpose of search and problem solving has been the emerging area of swarm intelligence. Honey-bees are among the most closely studied social insects. In this paper, an artificial bee colony algorithm is developed to solve clustering problems which is inspired by the bees' forage behavior. The ABC algorithm for data clustering can be applied when the number of clusters known a priori and are crisp in nature. To evaluate the performance of this algorithm, it is compared with other stochastic algorithms viz. ant colony, genetic algorithm, simulated annealing, tabu search and the hybrid K–NM–PSO algorithm. This algorithm was implemented and tested on several real datasets. The Preliminary computational experience is very encouraging in terms of the quality of solution found, the average number of function evaluation and the processing time required. There are a number of research directions that can be considered as useful extensions of this research. We can

combine it with some local search strategy or hybrid it with other meta-heuristic algorithms properly. Furthermore, applying the proposed algorithm to solve other optimization problems is also possible in further research.

## Acknowledgements

This work was supported by NSFC Major Research Program under Grants 60973089 and 60903009, Open Research Fund of the Symbol Computation and Knowledge Engineer of Education Ministry (93K-17-2009-K02) and the Special Fund for Fundamental Research of Central Universities of Northeastern University (90404015), and the National High Technology Research and Development Program of China (863 Program) (2009AA012122).

## References

Al-Sultan, K. S. (1995). A tabu search approach to the clustering problem. *Pattern Recognition, 28*(9), 1443–1451.

Bandyopadhyay, S., & Maulik, U. (2002). An evolutionary technique based on *K*-means algorithm for optimal clustering in RN. *Information Science, 146*, 221–237.

Basturk B. & Karaboga D. (2006). An artificial bee colony (ABC) algorithm for numeric function optimization. In *IEEE swarm intelligence symposium 2006, May 12–14*. Indianapolis, IN, USA.

Blake, C. L. & Merz, C. J. (1998). UCI repository of machine learning databases. Available from: <http://www.ics.uci.edu/-mlearn/MLRepository.html>.

Brucker, P. (1978). On the complexity of clustering problems. In M. Beckmenn & H. P. Kunzi (Eds.), *Optimisation and operations research. Lecture notes in economics and mathematical systems* (Vol. 157, pp. 45–54). Berlin: Springer.

Forgy, E. W. (1965). Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics, 21*, 768–769.

Goldberg D. E. & Deb, K. (1991). A comparison of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 69–93).

Güngör, Z., & Ünler, A. (2007). *K*-harmonic means data clustering with simulated annealing heuristic. *Applied Mathematics and Computation, 184*(2), 199–209.

Kao, Y.-T., Zahara, E., & Kao, I.-W. (2008). A hybridized approach to data clustering. *Expert Systems with Applications, 34*(3), 1754–1762.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing, 8*(1), 687–697.

Krishna, K., & Murty (1999). Genetic *K*-means Algorithm. *IEEE Transactions on Systems Man and Cybernetics B Cybernetics, 29*, 433–439.

Mualik, U., & Bandyopadhyay, S. (2002). Genetic algorithm based clustering technique. *Pattern Recognition, 33*, 1455–1465.

Murthy, C. A., & Chowdhury, N. (1996). In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters, 17*, 825–832.

Sander, J. (2003). *Course homepage for principles of knowledge discovery in data*. Available from: <http://www.cs.ualberta.ca/~joerg>.

Seeley, T. D., & Visscher, P. K. (1988). Assessing the benefits of cooperation in honeybee foraging: Search costs, forage quality, and competitive ability. *Behavioral Ecology and Sociobiology, 22*, 229–237.

Selim, S. Z., & Al-Sultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition, 24*(10), 1003–1008.

Selim, S. Z., & Ismail, M. A. (1984). *K*-means type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 6*, 81–87.

Shelokar, P. S., Jayaraman, V. K., & Kulkarni, B. D. (2004). An ant colony approach for clustering. *Analytica Chimica Acta, 509*, 187–195.

Sung, C. S., & Jin, H. W. (2000). A tabu-search-based heuristic for clustering. *Pattern Recognition, 33*, 849–858.

Teodorovic, D., Lucic, P., et al. (2006). Bee colony optimization: Principles and applications. In *Neural network applications in electrical engineering, 2006 (NEUREL 2006)* (pp. 151–156). Belgrade.